University of Glasgow

DEPARTMENT OF

AEROSPACE ENGINEERING

Unfactored Multiblock methods:

Part I - Initial Method Development

K.J. Badcock, S.Porter and B.E.Richards

Glasgow University Aero report 9511

Unfactored Multiblock methods:

Part I - Initial Method Development

K.J. Badcock, S.Porter and B.E.Richards

Glasgow University Aero report 9511

Unfactored Multiblock Methods:Part 1 Initial Method Development

K.J. Badcock, S.Porter and B.E.Richards [1]

Abstract. The use of implicit multiblock methods is discussed and an unfactored method based on a conjugate gradient type solution is described. Preconditioners appropriate to multiblock are considered along with future extensions to parallel computing.

1 Introduction

Simulation techniques are generally developed for geometrically simple problems such as the flow over an aerofoil or wing. This allows attention to be focused on the fundamental numerical problems such as the sharp resolution of shock waves, the accurate capturing of boundary-layers or minimising the computer time to obtain a solution. However, geometric complexity is present in most *real life* flow problems and especially so for those arising in the aerospace industry.For a numerical technique to be generally useful the extension to these problems must be tackled.

The first problem to be addressed when faced with a flow in a complex geometry is grid generation. It is not, in general, possible to generate a high quality structured single block grid for this type of problem. There are two methods which can be used to overcome this. Unstructured grids have sufficient flexibility to cope with any geometry without alteration (although the definition of the geometry poses difficulties in practice). However, solution techniques on unstructured grids are less advanced than those for structured grids. The multigrid acceleration technique is complicated by the lack of an obvious hierachy of grids which is present on a structured grid although various ways of overcoming this have been developed [3].

The most popular implicit method, alternating direction implicit (ADI) , is inapplicable due to the lack of an obvious ordering of the grid cells to define the alternating directions. Implicit methods based on iterative linear solution methods are promising because they require no essential changes from established structured grid methods since they typically only require a matrix-vector multiplication which can easily be implemented on an unstructured grid. However, the storage required is much larger on an unstructured grid due to the requirements of the connectivity information and can be prohibitive for large applications.

The alternative to an unstructured grid is a block structured grid (called multiblock). Here, the flow domain is split up into blocks and structured grids are generated in each block with grids in adjacent blocks being matched at common interfaces. The result is a good quality grid which is essentially structured.The major practical problem is the division of the geometry into blocks. Multigrid is well suited to multiblock because the updating is based on local information and hence the blocks can be treated independently except for cells adjacent to block boundaries where some minor inter-block communication is required. The same is true for implicit methods based on iterative linear solutions since the matrix-vector products require the same inter-block communication as an explicit step. However, preconditioning is required for effective convergence of the iterative solution and this is usually based on a direct linear solution which will be global in nature i.e. Gaussian elimination or back substitution across blocks will be required. This complicates both the serial and efficient parallel implementation of the multiblock method.

This report considers the construction of preconditioners which are appropriate for a parallel multiblock implementation. We consider making approximations to the preconditioner which decouple blocks. The only question which needs to be considered is how much these approximations degrade the convergence of the iterative linear solver since we are still solving the *exact* linear system to a prescribed tolerance by a conjugate gradient type method.

The report is organised as follows. First, the formulation of the problem is discussed with particular reference to decoupling the preconditioning between blocks. Results are then presented for convergence to a steady state of a transonic flow over an aerofoil. The behaviour as the number of blocks is increased and the effect of the block boundaries on the convergence is described. The effect of the sparsity pattern of the preconditioner is considered. Finally conclusions are drawn and further work discussed.

[1] Department of Aerospace Engineering, University of Glasgow, Glasgow, G12 8QQ, UK

Glasgow University Aerospace Engineering Report 9511.

2 Formulation

The essential features of the problem can be illustrated in one spatial dimension. Consider the solution of the convection equation

$$w_t + f_x(w)=0 \tag{1}$$

where f is the flux function of w, on the domain $0 \leq x \leq 1$.

To fix ideas we assume that the discretisation of the spatial term in equation 2 has a five point stencil. Adopting the notation that the approximations to w and $f_x(w)$ in the ith cell are written as u_i and

$$f_x(w) \approx R_i(u_{i-2},u_{i-1},u_i,u_{i+1},u_{i+2}) \tag{2}$$

then we can write one step of the explicit method with Euler local time stepping as

$$\mathbf{u}^{n+1}=\mathbf{u}^n - \Delta t_i \mathbf{R}(\mathbf{u}^n) \tag{3}$$

and one step of an implicit method as

$$(\mathcal{D} + \frac{\partial \mathbf{R}}{\partial \mathbf{u}})(\mathbf{u}^{n+1} - \mathbf{u}^n)= - \mathbf{R}(\mathbf{u}^n) \tag{4}$$

where $\mathbf{u}$ and $\mathbf{R}$ are vectors with the ith component given by u_i and R_i and D is a diagonal matrix whose ith component is given by $1/\Delta t_i$. In the present case with a five point stencil for the discretisation the Jacobian matrix of the discretisation $\partial \mathbf{R}/\partial \mathbf{u}$ is block penta-diagonal in structure and hence the updates coefficient matrix on the left hand side of equation 4 is also block penta-diagonal and we denote the five non-zero diagonal blocks at the ith line in the matrix as $\mathcal{W}2_i$, $\mathcal{W}1_i$, $\mathcal{C}_i$, $\mathcal{E}1_i$ and $\mathcal{E}2_i$. Here,

$$\mathcal{W}2_i=\frac{\partial R_i}{\partial u_{i-2}}, \mathcal{W}1_i=\frac{\partial R_i}{\partial u_{i-1}}, \mathcal{C}_i=\mathcal{D} + \frac{\partial R_i}{\partial u_i},$$

$$\mathcal{E}1_i=\frac{\partial R_i}{\partial u_{i+1}} \mathcal{E}2_i=\frac{\partial R_i}{\partial u_{i+2}}.$$

The linear system in equation 4 can be solved efficiently by (a) eliminating the blocks $\mathcal{W}2_i$ and $\mathcal{W}1_i$ by elementary row operations, for $1 \leq i \leq M$ where M is the number of grid points and (b) performing back substitution on the resulting matrix which is in upper triangular form. Note here that this solution process is sequential in nature since the elimination of the blocks $\mathcal{W}2_j$ and $\mathcal{W}1_j$ can only be accomplished once the blocks $\mathcal{W}2_i$ and $\mathcal{W}1_i$ have been eliminated for $1 \leq i < j$ and a similar problem is encountered during the back substitution. It is also global in nature since the updated solution in the ith cell depends on the updated values in all of the other cells.

The multiblock concept involves dividing the solution domain into blocks and generating grids and solving the problem in each block. For multi-dimensional problems this simplifies the task of generating high quality grids and makes a computer code general since the details of the problem are encoded in the inter-block connectivity and not in the computer code itself. The block decomposition also provides a natural partition of the problem for parallel processing. The efficiency of the parallel implementation will depend on the inter-block communication required by the solution algorithm. We shall consider this for our simple one-dimensional test problem. This is a contrived example since multiblock is used to deal with geometric complexities which do not arise in one-dimensional problems but this problem does contain the essential difficulties inherent for implicit methods in multiblock.

Consider a case with three blocks of equal size i.e block 1 contains the cells with $1 \leq i \leq M/3$, block 2 those cells such that $M/3 < i \leq 2M/3$ and block 3 has cells with $2M/3 < i \leq M$. We define the halo of a block as the set of cells which are required for the calculation of the updated solution but which are not members of that block. For the explicit method the halo of block 2 is the set of cells $\mathcal{H}=\{M/3 - 1, M/3, 2M/3 + 1, 2M/3 + 2\}$ ie the halo consists of the two cells adjacent to block 2 in each of the neighbouring blocks. For a parallel implementation when the blocks reside on separate processors, the communication required at each step corresponds to information relating to cells in the halo of each block. For the explicit method the values of the solution in the halo sets much be communicated before the residuals are calculated. Once the values in the halo sets of each of the blocks has been communicated to the processor on which the block resides the calculation for that block can proceed independently of the other blocks and hence processors.

However, let us now consider an implicit update. To calculate the matrix and the right hand side of equation 4, the inter-block communication is identical to that of the explicit step. However, the halo for each of the blocks consists of all of the cells in every other block due to the sequential nature of direct solution as discussed above. Hence, in parallel a direct solution will be inefficient since processors 2 and 3 will be idle whilst processor 1 carries out the elimination of the matrix elements $\mathcal{W}2$ and $\mathcal{W}1$ in block 1, then processors 1 and 3 will be idle whilst processor 2 carries out the elimination of the matrix elements $\mathcal{W}2$and $\mathcal{W}1$ in block 2 and finally, processors 1 and 2 will be idle whilst processor 3 carries out the elimination of the matrix elements $\mathcal{W}2$ and $\mathcal{W}1$ in block 3. Similar problems will arise during the back substitution.

A better potential way of solving the linear system is to use an iterative method which uses matrix-vector products as its main computational work. These products have the same halo as an explicit update and hence are well suited to parallel computation. Conjugate gradient methods are an example of this type of approach. However, conjugate gradient methods usually require preconditioning to be effective for the solution of linear systems which arise in practice and this preconditioning is provided by a direct solution of a simpler version of the linear system.

Hence, the preconditioning presents us with the main problem for an efficient parallel implementation of an implicit multiblock method with a linear solution based on a conjugate gradient type algorithm. We tackle this problem by considering ways of approximating the direct solution such that the halo of this part of the calculation is no larger than that of an explicit step. In the next section we describe an

approach to achieve this aim.

3 Blocked Direct Solvers for Multiblock

We consider the problem of approximating the solution to a linear system 4 arising from a multiblock grid such that the coupling between the blocks is minimised. Define the set of cells in the ith block of the grid as $\mathcal{B}_i$. Note that if we neglect all the matrix blocks

$$\frac{\partial R_i}{\partial u_j}$$

when $i \in \mathcal{B}_k$ and $j \notin \mathcal{B}_k$ then the solution of this reduced system decouples between blocks. The solution of equation 4 with this approximation, which we shall refer to as the blocked direct solution (BDS), is no longer exact and might not be good enough to maintain stability of the time stepping. However, this approximate solution could be used as a preconditioner for a conjugate gradient type solution of the exact system, resulting in an overall implicit method which has the same halo as an explicit step. The possible disadvantage of this approach is that, if the preconditioning is too approximate, the conjugate gradient iteration will converge too slowly.

To illustrate the method let us consider the solution of the linear system in a block which contains the cells {10,11,..,19,20}. The reduced linear system in this block is obtained by ignoring the Jacobian elements $\mathcal{W}2_{10}$,$\mathcal{W}1_{10}$,$\mathcal{W}2_{11}$,$\mathcal{E}2_{19}$,$\mathcal{E}1_{20}$,$\mathcal{E}2_{20}$.

This method was used in [6] for the ILU preconditioned solution of large sparse linear systems arising from an implicit method for the Euler equations discretised on unstructured grids. Methods were tried to compensate for the neglected terms by subtracting off their effect from the right hand side using values obtained from coarse grid approximations. However, the cost of forming the coarse grid approximation outweighed the benefits. A similar idea was used in [1] where with old values at block boundaries being used to subtract off the influence of the block coupling terms from the right hand side for the current time step. This was used for ILU preconditioning for QMR-CGS solutions. No indication was given as to how much this improved the performance of the preconditioning. An explicit treatment of block boundary terms was used in [2] together with Gauss-Seidel for interior points with a significant drop off in convergence being noted as the number of blocks increased.

Below we investigate these ideas for the extension to multiblock grids of work which has been carried out on implicit methods for single block grids.

4 Two-Dimensional Compressible Flow Equations

The presentation has so far been for a contrived one-dimensional problem. We are interested in being able to solve multidimensional fluid flow problems described by the Navier-Stokes equations. The implicit formulation is as described in eg [4] and the differences with the present work lie in the conjugate gradient solution of the linear system.

In the following we use the generalised conjugate gradient (GCG) method to solve the linear system as described in [5]. The preconditioning strategy which we shall adapt is based on an incomplete lower-upper (ILU) factorisation. The sparsity pattern of L and U is defined with respect to the sparsity of the unfactored matrix. The ILU factorisation can be decoupled between the blocks by making the BDS approximation for the unfactored matrix and we shall refer to this as Bl-ILU preconditioning.

We shall investigate three different sparsity patterns for the preconditioning. We map the rectangular grid in a block onto a vector of unknowns by rows in one of the coordinate directions. We shall define the sparsity pattern of the preconditioner with this ordering of unknowns by reference to the stencil this corresponds to on the mesh for each cell in the grid. We consider three cases as shown in figure 1. With increasing number these methods include more elements in the LU decomposition and hence should yield a more accurate approximation to the matrix inverse at the cost of increased calculation time in forming the preconditioner and increased storage. The potential benefits are a decreased number of GCG iterations to convergence and an increased robustness. Method 1 requires the calculation and storage of 36 elements per unknown in the linear systems, method 2 requires 52 elements and method 3 100 elements.

5 Results

5.1 Evaluation of Preconditioning Methods

As a test problem we shall consider the inviscid transonic flow over the NACA64A010 aerofoil at Mach 0.8 and zero incidence. Three multiblock meshes were constructed from a single block mesh which contains 2400 points. The multiblock meshes have 3, 6 and 12 blocks respectively.

First we present results to illustrate the behaviour of the three preconditioning methods. All of the results in this section were obtained on a grid with twelve blocks. The two numerical parameters which most effect the convergence of the linear solver and hence the overall convergence are the tolerance to which the linear solution is obtained and the CFL number which determines the condition number of the linear system at each implicit step, the higher the CFL number the more difficult the linear system.

The residual of the linear system is defined as

$$\frac{||Ax - b||_2}{||b||_2}.$$

The effect of the tolerance on the convergence to the flow solution is shown in figure 2 for method 1 and it can be seen that decreasing the tolerance from 0.1 to 0.001 makes the convergence smoother but has little effect on the convergence rate. The variation of time to convergence for the three methods with the tolerance at the optimum CFL number (ie the CFL number which yields the fastest convergence rate for that method) is shown in table 1. It is clear that the convergence time is insensitive to the tolerance but that decreasing

the tolerance increases the robustness. A tolerance of 0.001 is chosen for the remainder of this work.

The CFL number is a crucial parameter when iterative solvers are being used with an implicit method because of the balance that exists between minimising the the number of implicit steps by increasing the CFL number and reducing the number of iterations for the linear solver by reducing the CFL number. This balance is illustrated in figure 3. This plot shows the advantage of improving the preconditioning by using methods 2 or 3 as opposed to method 1. The difference between the convergence rate of methods 2 and 3 is small and the choice between these two methods must balance the increased memory requirement of method 3 as opposed to its increased robustness. To further allow comparison between the three methods the convergence of the flow solution at the optimimum CFL number is shown in figure 4 and it is clear that methods 2 and 3 easily outperform method 1.

The full flow solution convergence histories at a CFL number of 300 for the three methods are shown in figure 7. It is surprising that the terminal convergence rate is poor for method 1 compared with the other two methods and that there is an improvement for method 3 compared with method 2. The measure of convergence for the linear solution is based on the L2 norm of the residual and is hence a global measure of the error. However, for method 1 there remains some large local components of the relative error, even when the solution has converged by the global absolute measure. As the preconditioning is improved with methods 2 and 3 the local error is more evenly distributed and hence has a smaller effect on the convergence of the flow solver. Hence, using better preconditioning improves the robustness of the method.

5.2 Effect of Blocks

The second and main point of interest is the effect of the number of blocks on the convergence rate. Results are presented for methods 1 and 3 only since, from above, the performance of methods 2 and 3 is similar. The time to convergence for each method along with the average number of conjugate gradient iterations per implicit time step is shown varying with the number of blocks in tables 2 and 3 respectively. It is clear from table 2 that the number of CG steps per implicit iteration remains constant whilst increasing the number of blocks and hence blocking the ILU decomposition does not effect the quality of this preconditioner. The number of steps for method 3 increases with the number of blocks because the ILU decomposition is more accurate in each block and hence more is lost by neglecting terms at block boundaries. There is no trend for either of the methods with increasing CFL number. A strange feature of these results is that the number of implicit steps to convergence depends on the number of blocks with the 3 block case converging slowest at the lower CFL numbers. This behaviour requires further investigation.

Plots of the convergence to the flow solution at the optimal CFL numbers are shown in figures 5 and 6 respectively and confirm that the convergence rate is insensitive to the number of blocks. Further investigation for more blocks is required.

Plotting the pressure contours for the converged twelve block solution confirmed that the block boundaries have no influence on the solution. The shock wave on the upper and lower surfaces is located at block boundaries when 6 and 12 blocks are used. It is therefore surprising that the convergence of the linear solver for these grids is similar to that on the 3 block grid since the terms neglected for the preconditioning arise from a region where the flow is rapidly changing which will lead to large off-diagonal terms in the updates coefficient matrix.

6 Conclusions

Preconditioning methods based on the ILU decomposition have been investigated for multiblock implementation. An approximation which neglects terms which couple blocks was used to make blocks independent from the point of view of preconditioning. This will allow very efficient parallel application.

The main conclusions of this study are that:

- flow solution convergence is not strongly dependent on the number of blocks for a small test problem involving inviscid transonic flow, even when shocks are located at block boundaries
- decreasing the linear system tolerance improves robustness at little computational cost
- preconditioning methods 2 and 3 are superior to method 1 with method 3 being more robust than method 2 at the cost of increased storage.

Future work will focus on several issues:

- investigation of behaviour for larger meshes and a larger number of blocks
- inclusion of viscous effects and turbulence models
- investigation of flow acceleration methods and mixed discretisation models between blocks.

REFERENCES

[1] G.F.Carey A.K.Stagg, D.D.Cline and J.N.Shadid, 'Parallel, scalable parabolized Navier-Stokes solver for large-scale simulations', *AIAA Journal*, **33**, 102–108, (1995).

[2] C.B.Jenssen, 'Implicit multiblock Euler and Navier-Stokes calculations', *AIAA Journal*, **32**, 1808–1814, (1994).

[3] D.J.Mavriplis and A.Jameson, 'A multigrid solution of the two-dimensional Euler equations on unstructured triangular grids', *AIAA Journal*, **26**, 824–831, (1988).

[4] K.J.Badcock and B.E.Richards, 'Implicit time stepping methods for the Navier-Stokes equations', in *12th AIAA CFD conference, San Diego*. AIAA, (1995).

[5] O.Axelsson, *Iterative Solution Methods*, Cambridge University Press, 1994.

[6] V.Venkatakrishnan, 'Parallel implicit unstructured grid Euler solvers', *AIAA Journal*, **32**, 1985–1991, (1994).

Method	CFL. Number	0.1	0.01	0.001
1	300	5294	5478	5605
2	600	Blew Up	1188	1300
3	900	1340	1149	1428

Table 1. *The effect of the tolerance on convergence times for the three methods*

CFL.number	100	200	300
Number of blocks	-	-	-
3	11950	8350	5600
6	7432	5499	4056
12	7827	5550	4150

Table 2. *The effect of CFL. number and number of blocks on convergence times in work units for method 1*

CFL.number	300	600	900
Number of blocks	-	-	-
3	4004	1945	1117
6	2429	1358	1258
12	2843	1936	1428

Table 3. *The effect of CFL. number and number of blocks on convergence times in work units for method 3*

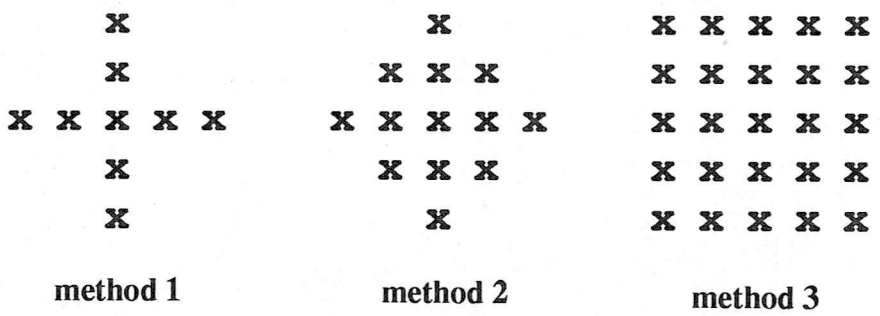

Figure 1. *Stencils for methods 1, 2 and 3.*

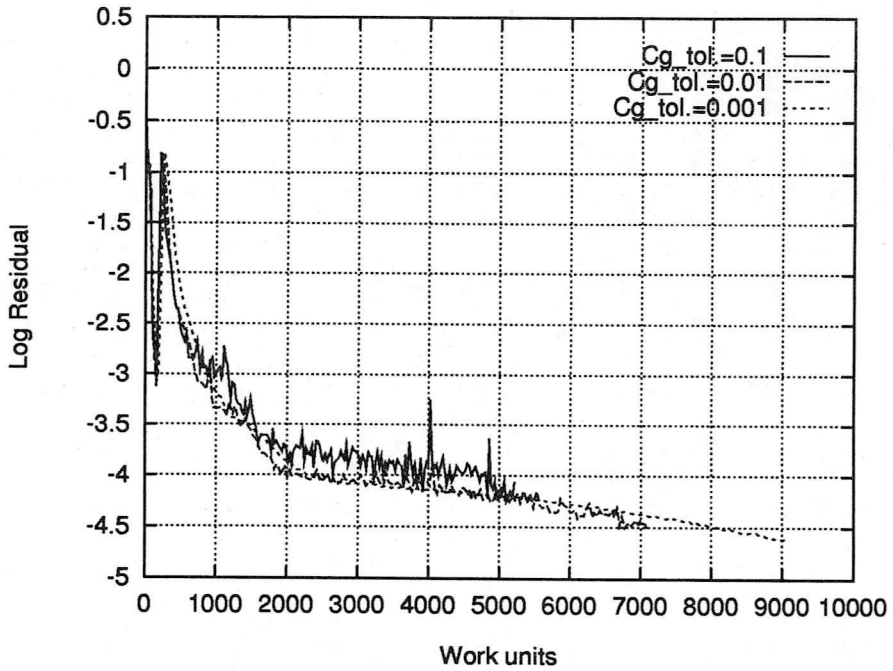

Figure 2. *Convergence of the flow solution on 12 blocks for method 1 at CFL=300 for varying tolerance.*

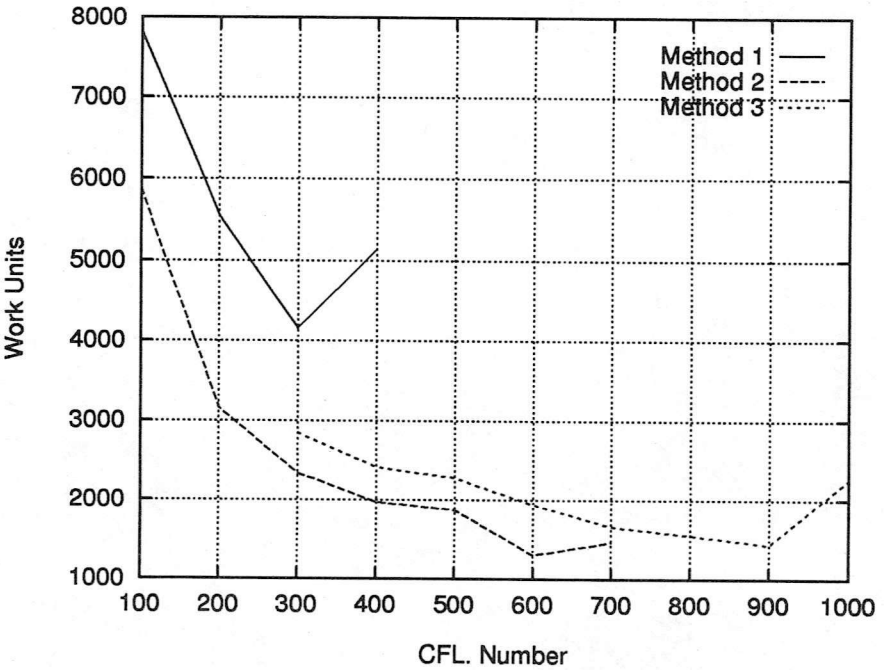

Figure 3. *Variation of time to convergence on 12 blocks with CFL number for methods 1,2 and 3.*

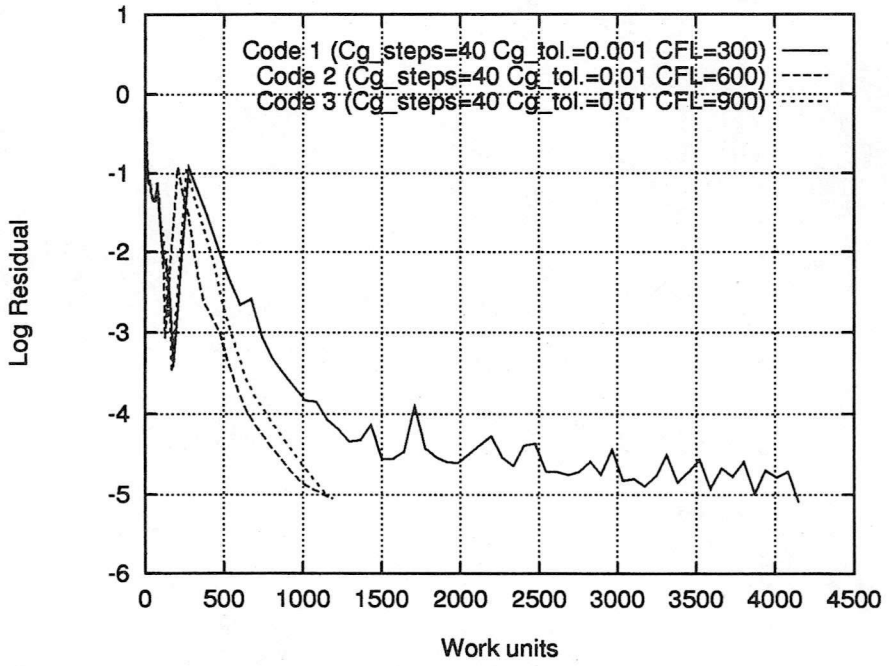

Figure 4. *Convergence of the flow solution on 12 blocks for the three methods at the optimal CFL numbers.*

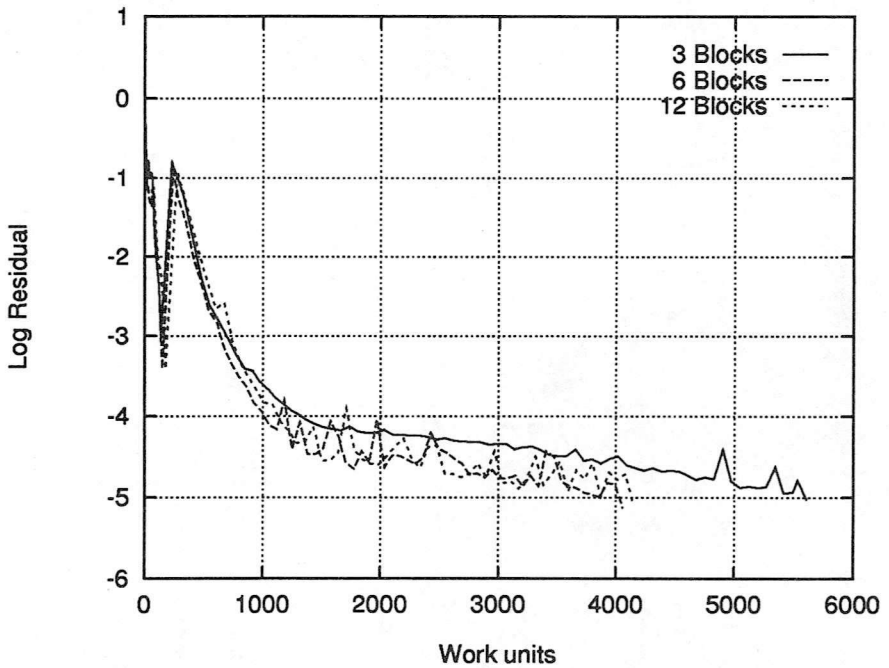

Figure 5. *Convergence of flow solver for 3,6 and 12 blocks at CFL=300 for method 1.*

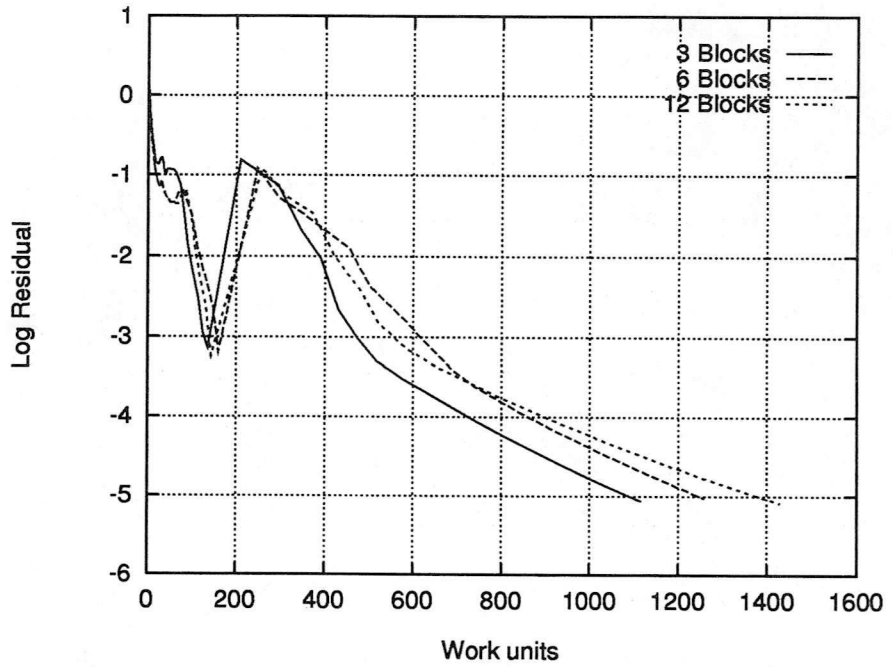

Figure 6. *Convergence of flow solver for 3,6 and 12 blocks at CFL=900 for method 3.*

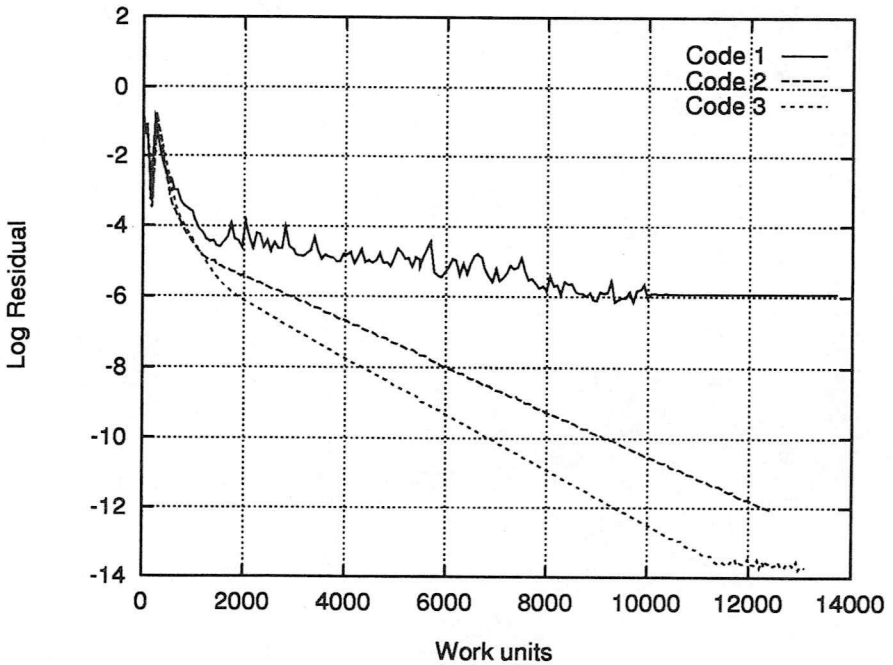

Figure 7. *Terminal convergence rates for the three methods.*